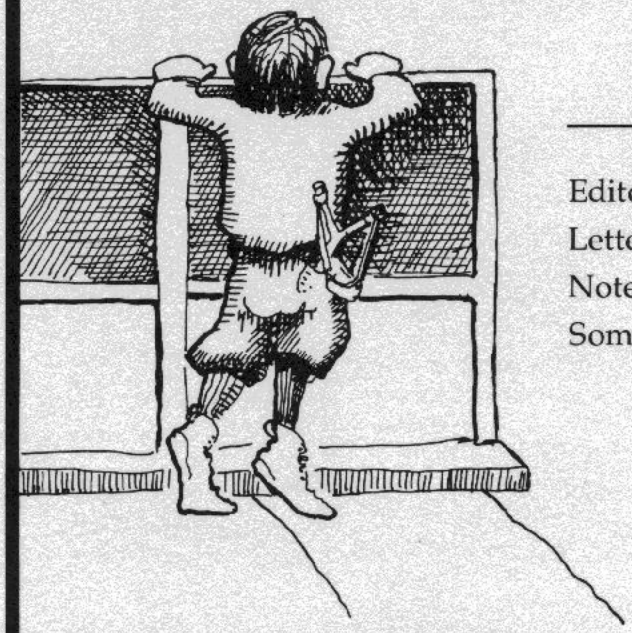
$3.00

MICRO CORNUCOPIA

JOURNAL OF THE BIG BOARD USERS

September 1981 No. 2

TABLE OF CONTENTS



REGULAR FEATURES

MICRO CORNUCOPIA
11740 N.W. West Road
Portland, Oregon 97229
503-645-3253

Editor & Publisher
David J. Thompson

Technical Editor
Ruth Fredine-Burt

Graphic Design
Sandra Thompson

Typography
Patti Morris & Martin White
Irish Setter

Cover Illustration
Gerald Torrey

MICRO CORNUCOPIA is published six times a year by Micro Cornucopia of Oregon, 11740 N.W. West Road, Portland, Oregon 97229.

SUBSCRIPTION RATES:

1 yr. (6 issues)	$12.00
1 yr. (Canada)	$15.00
1 yr. (other foreign)	$20.00

All subscription orders payable in United States funds only, please.

ADVERTISING RATES: Available on request.

CHANGE OF ADDRESS: Please send old label and new address.

SOFTWARE, HARDWARE, AND BOOK VENDORS: Micro Cornucopia is establishing a group of reviewers. We would very much like to review your Big Board compatible products for Micro C. Please send material to Review Editor, Micro Cornucopia.

WRITER'S GUIDELINES: All items should be typed, double-spaced on white paper or better yet, on disk. (Your disk will be returned promptly.) Payment is in contributor's copies.

LETTERS TO THE EDITOR: Please sound off.

MICRO CORNUCOPIA

Sept. 1981 **The Journal of the Big Board Users** **No.2**

There once was a Big Board so brisk.
It could eat all the bits off a disk.
It chewed up the bits, then spit out the pits,
which made feeding it software a risk.

Here We Go Again!

Exclusive!

What happens when a Xerox copies a Big Board? Why you get a "Worm", of course! That's right! The Xerox 820 is just a Big Board in disguise.

My informed sources say that last fall Xerox bought non-exclusive rights to manufacture a system based on the Big Board. Xerox re-laid out the board (4 layers) so that it would fit in the cabinet, they dedicated the SIO port B as a printer port, and they set up the disk interface (1771) to handle either 5 or 8 inch. Otherwise, it appears to be all Big Board, right down to the 2.5 MHz clock. The system PIO does the same things on both systems, bit for bit, according to Xerox's documentation.

Xerox had 50,000 orders in hand the day they shipped the first 820, and they expect to recoup all their startup costs by the end of this calender year. What a market for software and hardware developed around the Big Board. I'll say more about the 820 as information comes in. (I'd give my eye teeth to see a schematic and service manual for the 820.)

Picnic

We had a Saturday noon picnic to celebrate our first issue. It turned out that the Saturday we picked conflicted with every party/birthday/outing/etc. for three states around. But Sandy and I and those who came had six hours of very interesting and mellow conversation.

The knowledge, resources, and excitement among the local group members are terrific. I only wish all of you could have joined us.

The First Issue

Despite the speed of the U.S. Snail, a heartening number of readers have actually received issue no. 1. The responses from these lucky folks have made the daily trip out to our mailbox most enjoyable. The comments have included; 'surprised, happy, delighted '.

Though Micro C is a long way from being a success financially, feedback like this tells us that it is successful in other ways. We like doing it and we really appreciate your response.

Sometimes a dream generates momentum of its own. This one has.

Thanks.

David Thompson

David Thompson
Editor & Publisher

Letters

Dear Sir,

July came and July went by, and my mailbox has completely rusted out due to all that drooling.

Silly me! When I read 'Issue No. 1 will hit the streets during July' I assumed it was July 1981! But now I realize you meant July 1982. I'd better get a stainless steel mailbox or maybe not bother to wait, because the magazine will never get here.

Maybe it went the way of Mitt's Newsletter, the Digital Group Newsletter, and Processor Technology's "Access."

I hope not.

Joe Kish
758 Yucca Ridge Lane
San Marcos, CA 92069

Editor's note:

I called Joe; after all it was the least I could do for his mailbox. And besides, I think it's a great letter! (He did finally receive issue no. 1.)

Sandy and I made a desperate, last ditch effort to get all 500 first issues collated, bound, labeled, sorted and bundled in one afternoon so we could get the first issue in the mail on July 31. We missed the 8 PM deadline at the post office by 15 minutes.

So the magazine was mailed Monday morning, August 3rd. (So much for hitting the streets in July.)

Someday maybe I'll write a book about starting a users group magazine. I could almost write the book about the first issue, and Murphy would certainly be a leading figure. (For those of you who don't know Murphy, he is the one credited with the first voyage of the Titanic.)

Quote from Murphy:

If there is no way
your plan can fail,
you simply don't have
all the information.

Dear Editor,

I bought a bare board version and built it up from scratch. I had to buy about $80.00 worth of parts beyond what I had around. I have it up and running CP/M and am currently working on packaging it in a terminal-type case with a Ball Brothers CRT. The unit is going to be used for text processing and formatting for a friend's photo typesetter. My other computer is an LSI-11 and I also use

(continued next column)

Supporting A Language

By David Thompson

Throughout these early months of Micro Cornucopia, I have been looking at commercial and public versions of various languages with the hope of finding a semiofficial language for this group.

A common high level language would mean we could pass around source code in something other than assembler. But the language would need to be powerful enough for substantial commercial applications and inexpensive enough that most of the people in the group could afford it.

Letters continued

my H19 with the DEC-20 at work. I think the Big Board is an excellent value and very useful.

I agree that Frank Gentges' idea about the parallel ports is excellent. That would take care of most of the board's limitatons. I think your publication has already been worth the price and I suspect that an active users group with a publication will enhance the usefulness of the hardware significantly.

Doug Faunt
PO Box 11142A
Palo Alto CA 94306

Dear David,

CONGRATULATIONS!!! FANTASTIC!!! You really made it. It looks great and reads great. You are certainly to be congratulated for undertaking such a task that should be helpful to so many.

I hate to mention that Momma and I are just back from five weeks vacation in the Smokey Mountains in Tennessee. I am about ready to get my feet on the ground again. I hope that I can get back on track to help keep the pipe full of articles for future issues.

Don Retzlaff
6435 Northwood
Dallas TX 75225

Editor's note,

What can I say? Thanks again Don, without you and John Jones and Andrew Beck, and the rest of you who are writing up things for future issues this wouldn't be possible. (As for the five whole weeks in the Smokey Mountains, that's just not fair.)

■ ■ ■

Plus, it would need to produce fast and compact object code, encourage readable source code, and promote structured programming. (Whew!)

I am looking seriously at three languages: Forth, Pascal, and C. Of these three, C is presently leading. One reason is that all the versions I have seen have been upwardly compatible with Bell Lab's C.

Versions of C that I'm aware of:

Small C (Public)
Small C+ (Public)
Tiny C ($100)
CW/C ($75)
BDSC ($145)
Supersoft C ($200)
Whitesmith's C ($600)

(The prices are approximate.)

Whitesmith's C is a full blown version of the language. In fact, sources tell me that it was created by three fellows who worked on C for Bell Labs. They left Bell in order to develop and market C for the business and scientific community.

I've heard that BDSC is a competent enough subset to be an option for someone writing commercial applications. It has its own users group and publication. All this for $145, such a deal. (Lifeboat is offering discounts on quantity purchases of BDSC.)

CW/C is an expanded version of Small C with lots of nice utilities, but I don't know if it is ready to do commercial work. However, it still looks like quite a bargain at $75.

Tiny C is the only interpreter in the bunch. It also comes in compiler form for about $300. The only thing I have heard about Tiny C is that it has an excellent manual (and I heard that fourth or fifth hand).

Supersoft's C is new on the market. The ads say that they support 'most' of version 7 Unix. If that includes floating point and pointer arithmetic, then it would be a very credible piece of of software, assuming they have taken time to exorcise bugs.

The standard text on C is:

"The C Programming Language"
by Kernighan and Ritchie
Prentice-Hall

■ ■ ■

Parallel Print Driver

By John P. Jones
5826 Southwest Ave.
St. Louis, MO 63139

This is a simple parallel printer driver that can be incorporated into any CP/M BIOS.

On first entry, the program initializes PIO port B and the interrupt vector register. The program also modifies the BIOS jump table so that all subsequent calls for list output bypass the initialization routine.

As each character is output to port B, a flag byte is set, indicating that the printer is busy. When the printer is again ready, the PIO does an interrupt. The sole purpose of the interrupt service routine is to reset the 'printer busy' flag. The character output routine tests the flag byte and loops until it is reset. When the flag is reset, a character is sent and the flag is again set.

■ ■ ■

Parallel Print Driver Listing

```
;
;
;
          ORG     CBIOS
;
          JP      BOOT            ;STANDARD JUMP TABLE TO
BVECTR:   JP      WBOOT           ;THE SUBROUTINES OF CBIOS
SVECTR:   JP      CONST
IVECTR:   JP      CONIN
OVECTR:   JP      CONOUT
          JP      OPNPRT          ;LIST DEVICE VECTOR
          JP      CONOUT          ;PUNCH DEVICE VECTOR
          JP      CONIN           ;READER DEVICE VECTOR
          JP      HOME
          JP      SELECT
          JP      SEEK            ;LIST DEVICE STATUS VECTOR
          JP      SETSEC
          JP      SETFTR
          JP      READ
          JP      WRITE
          JP      CONST
          JP      TRANS
;
OPNPRT    EQU     $               ;PORT B=OUTPUT
          LD A,   0FH             ;PIO/B CONTROL PORT
          OUT (0BH),A             ;INTERRUPT VECTOR B
          LD A,   1CH             ;LOAD VECTOR REGISTER
          OUT (0BH),A             ;ENABLE INTERRUPTS
          LD A,   87H
          OUT (0BH),A
;
          PUSH    HL              ;INTERRUPT DEST ADDR
          LD HL,  PIOINT          ;STORE AT VECTOR
          LD (0FF1CH),HL          ;CP/M ENTRY
          LD HL,  PRTCHR          ;ALL SUBSEQUENT ENTRIES SKIP INIT
          LD (OVECTR+4),HL
          POP     HL
          LD A, 0FFH              ;NON-ZERO TO A
          LD (INTPND),A           ;DECLARE INTERRUPT NOT PENDING
;
;
PRTCHR    LD A, (INTPND)          ;INTERRUPT PENDING?
          OR A
          JR Z, PRTCHR            ;IF YES, WAIT TIL NOT
          XOR A                   ;ZERO TO A
          LD (INTPND), A          ;SET PENDING AGAIN
          LD A,C                  ;GET CHAR
          CPL                     ;PRINTER NEEDS INVERTED DATA
          OUT (0AH), A            ;SEND CHAR
          RET
;
INTPND    DEFS    1               ;INTERRUPT PENDING FLAG
;
PIOINT    PUSH    AF
          LD A,   0FFH            ;ANY NON-ZERO OK
          LD (INTPND), A
          POP     AF
          EI
          RETI
```

end

Notes From Garland, Texas

By David Thompson

Clearing up the screen.

The clear-to-end-of-screen command is CONTROL Q, not CONTROL W as indicated in the documentation.

Bringing up stubborn boards.

A number of people have been contacting Jim and me about problems they are having bringing up boards. One of the most common symptoms is a pattern of two characters on the screen or a screenful of random garbage. Either way, it basically means that the board probably didn't finish loading the PFM monitor in RAM so it could try to clear the screen.

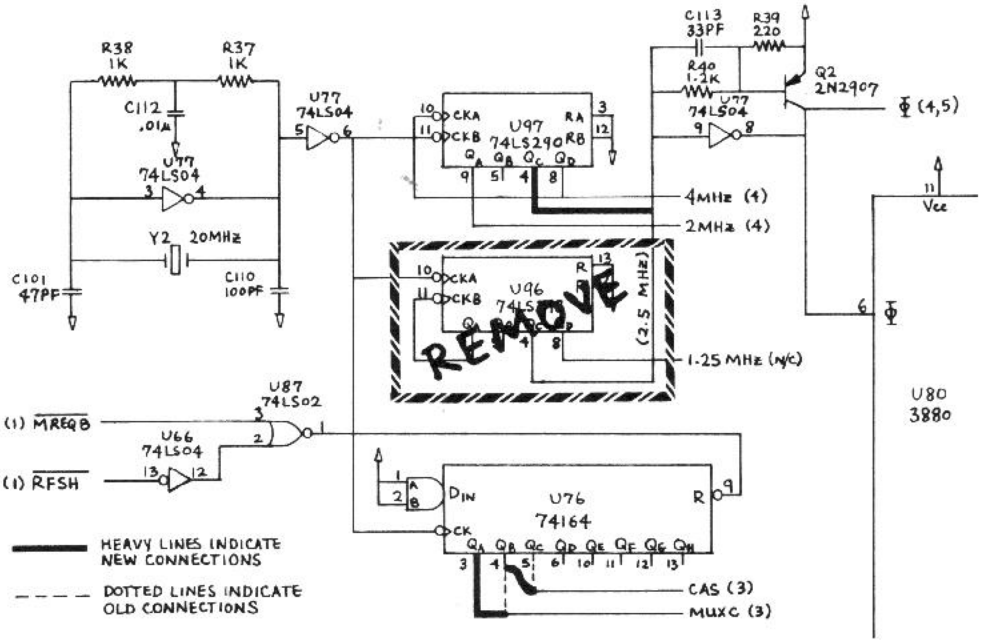
Jim is going to put together information about what they look for when they troubleshoot boards. Hopefully, I will have that in time for the next issue.

Don't forget the 90 day guarantee which completely covers defective parts and boards. Plus, he has been doing out-of-warranty or pilot error repairs very reasonably. Most of the time these charges have been between $25 and $50. The maximum so far has been $75 (the board had to be almost completely resoldered, among other things). That's pretty hard to beat.

Two CP/Ms

I have noticed that some software which runs on one Big Board system will not necessarily run on another. I also noticed that there are two different IDs when CP/M boots.

I called Jim about this and he said that those folks who used the BIOS he sent out with the boards and who did their own incorporation into CP/M have a version which origins the BIOS at EA00. All the folks who bought CP/M already modified for the Big Board have a BIOS starting at E800. The difference has led to some problems with software which depends on having BIOS in a certain place.



4 MHz Modification Version 2

Jim said the ready-to-run version has BIOS shifted down 200H because they thought they needed room to store 256 bytes (a double-density sector) in high memory. Then the data could be moved into low memory in 128 byte chunks and accessed. Jim isn't sure whether there is going to be a use for this space but he is concerned that we maintain consistancy.

According to Jim, it's easy to make the EA00 BIOS into an E800 BIOS.

Original—.RES.(MSIZE-20)*1024
New—.RES.((MSIZE-20)*1024)-200

Now reassemble the mess and you too can ORG at E800.

By the way, a pretty reliable way to tell which version you have is to look at the ID that's displayed when you boot CP/M. If it just says "60k CP/M version 2.2" then you probably ORG at EA00. If the prompt includes the words "BIG BOARD" then you already ORG at E800.

The separate BIOS (and monitor etc.) disk Jim is shipping with orders now ORGs at E800. If you would like the latest version rather than reassembling BIOS with the modification above, send Jim a disk and $3.00 for shipping.

4 MHz (Again).

This is an updated version of the 4 MHz mod printed in issue no. 1. This version reportedly does not require special ram. Jim says he has 300ns 4116 working consistently using this mod. The only difference between this one and the previous one is that the CAS and MUXC lines are each moved left one pin on U76 (shift register) so that they change states 50ns earlier. This change means that the system meets the precharge requirements for the slower RAM.

4 MHz Mod Version 2

1. Cut the trace (bottom of the board) to U76 pin 4.
2. Connect the cut trace (MUXC) to U76 pin 3.
3. Cut the trace (bottom of the board) to U76 pin 5.
4. Connect the cut trace (CAS) to U76 pin 4.
5. Remove U96.
6. Connect U97 pin 4 to U96 pin 4.
7. Don't replace U96.

(continued next page)

CP/M patch for serial printer port.

This CP/M modification redirects the list device output to serial port B. The default data rate is 300 baud. This patch does not force the Big Board to poll any of the handshake lines on port B. Thus, it has no way of knowing if the printer buffer is full. (May or may not be a problem.) This modification is for those who ORG at E800.

Enter the characters inside the quotation marks. <CR> = carriage return.

The patch:

1. Power up the Big Board (BB).
2. Place a CP/M disk with SYSGEN on it, in drive A.
3. Boot CP/M.
4. Enter "SYSGEN" "<CR>"
   Displays: SYSGEN VER. 2.0
   Displays: SOURCE DRIVE NAME . . .
5. Enter "A"
   Displays: SOURCE ON A, THEN TYPE RETURN
6. Enter "<CR>"
   Displays: FUNCTION COMPLETE . . .
7. Hit the BB RESET switch <CR>

NOTE: You now have an image of Boot, CP/M, and Bios in RAM starting at 0900H.

8. Remove the source disk from drive A.
9. Enter "M22C7" "<CR>"
   Displays: 22C7 00
10. Enter "79"
11. Enter "C3"
12. Enter "18"
13. Enter "F0"
14. Hit spacebar to return to PFM.
15. Enter "M1F90" "<CR>"
16. Enter "47"
17. Enter "EB"
18. Hit spacebar to return to PFM.
19. Place blank disk in drive A.
20. Enter "G100"
    Displays: SYSGEN VER 2.0
21. Enter "<CR>"
    Displays: DESTINATION DRIVE . . .
22. Enter "A"
    Displays: DESTINATION ON A . . .
23. Enter "<CR>"
    Displays: FUNCTION COMPLETE . . .
24. Enter "<CR>"

The disk now contains a CP/M system that supports CONTROL P (and PIP LST:=) for listings. As mentioned above, the output is on serial port B and is 300 baud.

Editor's note:

To change the baud rate, create F.COM as follows:

1. Enter "DDT" "<CR>"
2. Enter "A100" "<CR>"
3. Enter "MVI A,XX" "<CR>"
4. Enter "OUT 0C" "<CR>"
5. Enter "JMP 0" "<CR>"
6. Enter "<CR>"
7. Enter "G00" "<CR>"
8. Enter "SAVE 1 F.COM" "<CR>"

This routine sends a single byte (XX) to the channel B baud rate generator. I am working at 9600 baud so I replace XX with 0E. See the Big Board Theory of Operation for other baud rates.

Once you have completed the baud rate program, simply enter "F" "<CR>" from the CP/M prompt to set the baud rate.

No UPS to a PO Box?

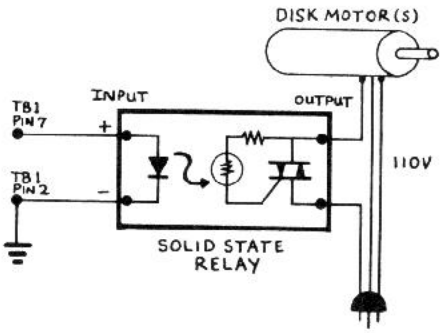
Jim Tanner lists his mailing address as a PO Box but he also has a street address that works for both the post office and United Parcel Service. (The ZIP is different.)

Jim Tanner
Digital Research Computers
2702 Industrial Lane
Suite J2
Garland, Texas 75041
Phone 214-271-3538

■ ■ ■

Disk Drive Motor Control

By David Thompson



Disk AC Control Circuit.

If you're tired of listening to your disk drives grind on hour after hour, here's relief.

The board must have the timer option installed and you must jumper pin 3 to pin 4 and pin 7 to pin 8 on JB2. This supplies the one second interrupt to the Z80. If the Z80 counts all the way to 30 after the most recent disk access then it sends a command to the system PIO to drive the output of U112 pin 2 low.

Terminal 7 on the Big Board power connector is tied to U112 pin 2. This terminal is high (about 4V) when the system is doing a disk access and goes low if there hasn't been an access for 30 seconds.

Simply connect the input of an optically isolated solid state relay between terminal 7 and ground. Then connect the output in series with the AC to the disk drive motors. (But do not connect in series with the drives' DC supply.)

I tried mechanical relays at first, but even the type made to be driven by TTL have problems. Whenever you use mechanical switches to start and stop motors you get interesting transients on the AC line. Interesting transients occasionally cause CPUs to go off picking daisies.

I am now using an ITT solid state relay P6-3DCC-120R5. It has a (P6) package, a 3VDC (3D) input, a 120VAC output with random switching point (120R), and it handles up to (5) amps. It is also small, quiet, and hasn't yet sent the system packing.

■ ■ ■

Jumpering The Wild Shugart

By David Thompson

Shugart set a new standard for obscurity when they came out with their SA 801 user's manual.

It's not that they don't tell you how to jumper their drives, the only problem is figuring out what they told you. Once you figure it out, don't go back and look at the manual, you'll just get confused again.

So on that note, here's what I figured out.

For drive A, jumper only the following: DC, C, DS1 (Drive Select 1), T2, T3, T4, T5, T6, HL, A, B, T1, 800, Y.

For drive B, change DS1 to DS2. For drive C, change DS1 to DS3, and so on.

For the last 9 months or so, Shugart has been shipping drives with a new circuit board. The new board is completely interchangeable with the old one, but the new one does not use the -5/-15V pin on the DC supply jack (J5). The pin is there but is not connected to anything because the new board does not need -5V.

One way to tell whether you have a new or old style drive is to check the bottom left hand corner on the circuit board. The old drive has a -5V regulator there. On the new one, that corner is pretty empty. Also, the resistance from the -5V pin to ground is infinite on the new boards.

I had one of the new boards but the old documentation so I spent a couple of 'interesting' evenings trying to make sure the -12V I was supplying would be properly turned into -5V on the board. (Oh well, if everyones' documentation were perfect there probably wouldn't be so much need for user groups.)

Note: The following information is from Bill Klevesahl, Shugart's product manager for the SA 800 series.

Test points for both boards.

1,2 Amplified read signal
5,6,7 Ground
10 -Index
11 +Head Load
12 -Index and Sector Pulses
16 +Read Data
25 +Write Protect
26 +Detect Track 0
27 +Step Pulse

Test points on the old board only.

3,4 Differential Read Signal (this signal is now hidden inside the new LSI read chip).
21,24 -Data Separator Timing (there is no longer a pot to adjust this).

Test points on the new board only.

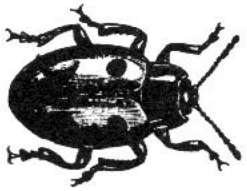
8 +Data Window (for checking FM data separation).

Optional features on the new board.

- Add-trace option TS enables true FM data separation, maintaining synchronization during address marks.
- Add-trace option NFO prevents the head from being forced out past track 0.

■ ■ ■

BUG



The formatting program listed in issue 1 contains a bug. If the program has a problem accessing a disk in drive B, it reformats the disk in the default drive (A).

Issue 3 will include a revised format program.

Coming Up

Articles you'll be seeing in the future.

- Reverse video cursor
- 5 inch disk interface
- Real time clock routine
- Converting a TV into a real video monitor
- More on the PFM monitor
- Review of 3 assembly language texts
- Bios modifications

Articles we'd love to see.

- Trials and tribulations of bringing up a Big Board
- How you've improved the PFM monitor
- Hard disk interface
- Filling out the second bank with system RAM
- DMA interface
- Double density disk interface
- A graphics display
- A speech generator
- A simple ROM burner
- Interfacing with particular printers etc.
- An in-depth series on CP/M
- Reviews of FIG Forth and Forth 79
- Reviews of BDSC, Whitesmith's C, CW/C and Supersoft's C
- Computer consulting using a Big Board
- Reviews on peripherals, keyboard, video monitor, power supply, cabinet, disks, etc.
- Other software reviews. Even if you are just borrowing a copy to evaluate, please let us know how you like it.
- Book reviews

If you are immersed in any of these projects, please share your experience with all of us.

Direct Input Routine

By Andrew P. Beck

AB Computer Products
PO Box 571
Jackson, NJ 08527

Assembly Listing

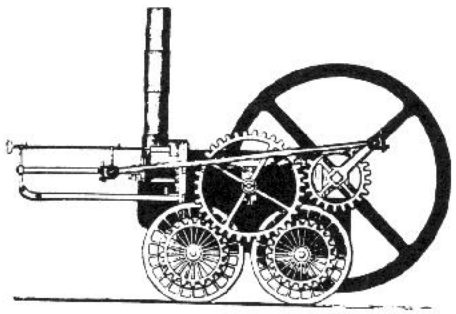
```
F800    E5       SUBR    PUSH HL       ;SAVE ADDRESS OF HL%
F801    CD06F0           CALL KBDST    ;GET KBD STATUS
F804    B7               OR A          ;IF A=0 DATA AVAILABLE
F805    CA0EF8           JP Z ISDATA   ;JP TO DATA SAVE ROUTINE
F808    E1               POP HL        ;GET ADDRESS BACK
F809    3C               INC A         ;A=FF IS NO DATA, MAKE IT 0
F80A    77               LD (HL),A     ;STORE 0 IN HL%
F80B    23               INC HL        ;DO BOTH BYTES
F80C    77               LD (HL),A
F80D    C9               RET           ;RETURN WITH HL% = 0
F80E    CD09F0   ISDATA  CALL KBDIN    ;GET INPUT CHAR INTO A
F811    E1               POP HL        ;GET ADDRESS OF HL% BACK
F812    77               LD (HL),A     ;STORE DATA, LOW ORDER
F813    23               INC HL
F814    3600             LD (HL),0     ;HIGH ORDER = 0
F816    C9               RET           ;RETURN TO BASIC
```

```
-- Poke the above program into F800+ --

500 SUBR = &HF800
510 DATA &HE5,&HCD,&H06,&HF0,&HB7,&HCA,&H0E,&HF8
520 DATA &HE1,&H3C,&H77,&H23,&H77,&HC9,&HCD,&H09,&HF0
530 DATA &HE1,&H77,&H23,&H36,&H00,&HC9
540 FOR I=0 TO 22
550 READ INST
560 POKE SUBR+I,INST
570 NEXT

-- Demonstration routine --

580 HL%=0
590 CALL SUBR (HL%)
600 IF HL%=0 GOTO 590
610 IF HL%=3 THEN STOP
620 PRINT CHR$(HL%);
630 GOTO 590
```



This routine makes it possible to do direct input with Microsoft basic. First, a machine language subroutine is poked into an unused area of the system monitor.

This subroutine calls the monitor subroutine and the monitor checks to see if an input character is available. If none is available, the HL% is set to zero. If a character is available, it is stored in HL% before a return is executed.

In the demonstration program, a returned character is echoed on the console. If the character is ^C, the demonstration stops.

■ ■ ■

More Power Supplies

By David Thompson

I just received a catalog from ACDC Electronics and they list a power supply that should power the Big Board and a couple of drives. (Like the Power One, you still have to finagle +12V but that isn't hard, see Issue no. 1.)

Model ETV801 provides:

+5V at 9 amps
-12V at 0.8 amps
+24V at 4.5 amps peak

Price is $132 (list, single)

They don't mention how they handle over-current protection, but they do indicate that they only have over-voltage protection on the +5V line unless you specify the -1 option. They don't say how much extra you pay for the option.

ACDC Electronics
401 Jones Rd
Oceanside, CA 92054

Power/Mate also has an open frame linear with the same specifications as the ACDC model above, but the PowerMate model ED-132AV lists for $120 (single).

Power/Mate
514 S River St
Hackensack, NJ 07601

■ ■ ■

Something New

DataCast
345 Swett Road
Woodside, CA 94062

I just received issue no. 1 of DataCast and I'm impressed, very impressed. This is a bimonthly magazine for 'major micro systems and telecommunications.' 'Major micro systems' means CP/M in a business or OEM environment and 'telecommunications' means networking.

Jim Warren, guiding force behind the West Coast Computer Faire, is behind this magazine and I suspect it will be around for a long while. Subscriptions are $18 per year (6 issues).

He is starting with a staff of 19 (if you include the mascot, Sir Lick-A-Lot) and it shows. The first issue is 64 pages and about 60 pages of that is copy.

Some first issue articles:

- What is Telidon and Why is AT&T Adopting It?
- Overview of Home Information Services
- A Seminar for Independent CP/M Software Vendors
- Software Documentation Protocols
- An Index to CP/M Software and Vendors

Other Interesting Periodicals

Dr. Dobb's Journal
PO Box E
Menlo Park, CA 94025

Lifelines
1651 Third Ave
New York, NY 10028

Please let us know about your favorite magazines.

Program Storage Above PFM

By Don Retzlaff

6435 Northwood
Dallas, TX 75225

There are numerous times when you want to write a small assembly language program to use as a printer driver or other routine. These small utilities need to reside in high memory so they can operate at the same time as routines which reside in the normal transient program area (starting at 0100H).

Since programs are loaded starting at 0100H, these utilities must load themselves into high memory.

There is a considerable amount of memory available above PFM that is not dedicated to any other use. PFM version 3.3 uses upper memory starting at F000H through F7E6H. The RAM area FF00H through FFC8H is used for data storage. This leaves the memory from F7E7H through FEFFH and FFC9H through FFFFH available for your use. Not all of this space is really available since future releases of PFM could use some of this space.

I recommend that you limit your programs to the following areas: (FA00H through FEFFH and FFE0H through FFFFH).

Moving the program up

In order for your routine to start out as a normal COM file but wind up in upper memory, it has to do a quick shuffle.

1. When the COM file is executed it is loaded into memory starting at 0100H.
2. Execution starts at 0100H.
3. The first few statements (starting at 0100H) must copy the routine into upper memory.
4. An initialization routine may then be executed.
5. Control is then transferred to the routine or back to PFM.

In order to accomplish all of the above it is necessary to do the following:

1. Write your assembly language routine as follows:
   a. The origin is set at the desired point where your routine is to reside.
   b. Your program must start with a short move routine.
   c. An initialize routine usually follows that patches (hooks) your routine into the monitor or PFM.
   d. Your routine follows.
   e. The last statement defines the length of the program.
2. Assemble your program.
3. Execute DDT and load your HEX file into memory. Typically this is done as follows:

   >A DDT NAME.HEX

This will load your program into memory at the desired location (example EA00H). The program will not execute.

DDT will print out starting and ending addresses.

NEXT PC/n
FAxx FA00

4. Using DDT, move the program from upper memory to 0100H.

   MFA00,FAxx,0100

5. Transfer control back to PFM by typing:

   G0

6. Save the program using the SAVE command.

   SAVE 1 NAME.COM

You must save the program in 256 byte blocks. Using '1' will save 256 bytes, '2' would save 512 bytes, etc.

7. The program is now ready for execution as a COM file.

The above procedure may seem long and rather involved but after you have done it a few times you will find it very quick and simple.

■ ■ ■

PFM Monitor Listing (continued from issue no. 1)

```
F37B C0          0745          RET     NZ          ;ERROR IF > 4 NUMBERS ENTERED
F37C C5          0746 PARA2:   PUSH    BC          ;SAVE PARAMETER COUNT
F37D CD9FF3      0747          CALL    GETHEX      ;READ A NUMBER FROM LINE BUFFER
F380 C1          0748          POP     BC
F381 D8          0749 PARA4:   RET     C           ;ERROR IF RESULT OVER 16 BITS
F382 DD217CFF    0750          LD      IX,PARAM1   ;POINT TO PARAM STORAGE AREA
F386 DD09        0751          ADD     IX,BC       ;ADD PARAMETER COUNT IN BC
F388 DD7500      0752          LD      (IX+0),L
F38B DD7401      0753          LD      (IX+1),H    ;STORE DATA RET FROM 'GETHEX'
F38E FE20        0754          CP      ' '
F390 28E4        0755          JR      Z,PARA1-$   ;GET ANOTHER ITEM IF SPACE
F392 FE2C        0756          CP      ','
F394 28E0        0757          JR      Z,PARA1-$   ;GET ANOTHER ITEM IF COMMA
F396 FE0D        0758          CP      CR
F398 37          0759          SCF                 ;ELSE CHECK FOR CARRIAGE RETURN
F399 C0          0760          RET     NZ          ; AND EXIT WITH CY=1 IF NOT
F39A 79          0761 PAREND:  LD      A,C

                                ADD     A,90H
                                DAA
                                ADC     A,40H
                                DAA
                                JP      OUTPUT
                                ;
                                ;PMSG PRINTS THE STRING OF ASCII CHARACTERS
                                ;POINTED TO BY THE RELATIVE ADDRESS IN DE
                                ;UNTIL AN EOT IS ENCOUNTERED IN THE STRING.
                                ;
F3E3 C690                       EOT     EQU     04H
F3E5 27                         CR      EQU     0DH
F3E6 CE40                       LF      EQU     0AH
F3E8 27                         ;
F3E9 C315F4                     PNEXT:  EX      (SP),HL
                                CALL    PMSG
                                EX      (SP),HL
0004                            
000D
000A
F3EC E3
F3ED CDF2F3
F3F0 E3
```

```
F39B  CB3F      0762          SRL     A           ;A=COUNT OF NUMBERS ENTERED
F39D  3C        0763          INC     A
F39E  C9        0764          RET
                0765 ;
                0766 ;GETHEX CONVERTS ASCII TO BINARY AND DOES
                0767 ;HIGH LIMIT CHECKS TO LESS THAN 17 BITS.
                0768 ;CARRY SET ON ILLEGAL CONVERSION RESULT
                0769 ;TERMINATING CHARACTER RETURNS IN A.
                0770 ;HL RETURNS WITH 16 BIT BINARY INTEGER
                0771 ;
F39F  210000    0772 GETHEX: LD      HL,0
F3A2  180B      0773          JR      GNUM3-$
                0774
F3A4  0604      0775 GNUM1:  LD      B,4
F3A6  29        0776 GNUM2:  ADD     HL,HL       ;MULTIPLY RESULT BY 16
F3A7  D8        0777          RET     C           ;RETURN IF IT OVERFLOWS 16 BITS
F3A8  10FC      0778          DJNZ    GNUM2-$
F3AA  5F        0779          LD      E,A         ;APPEND NEW LOW ORDER DIGIT
F3AB  1600      0780          LD      D,0         ;AND GET RESULT BACK INTO DE
F3AD  19        0781          ADD     HL,DE
F3AE  D8        0782          RET     C           ;RETURN IF OVERFLOW
F3AF  FD7E00    0783 GNUM3:  LD      A,(IY+0)    ;GET A CHAR FROM LINE INPUT
F3B2  FD23      0784          INC     IY          ; BUFFER @ IY AND BUMP IY
F3B4  4F        0785          LD      C,A
F3B5  CDBDF3    0786          CALL    ASCHEX      ;CONVERT ASCII TO NUMERIC
F3B8  30EA      0787          JR      NC,GNUM1-$
F3BA  79        0788          LD      A,C
F3BB  B7        0789          OR      A
F3BC  C9        0790          RET
                0791 ;
                0792 ;
F3BD  D630      0793 ASCHEX: SUB     '0'
F3BF  D8        0794          RET     C
F3C0  FE0A      0795          CP      10
F3C2  3F        0796          CCF
F3C3  D0        0797          RET     NC
F3C4  D607      0798          SUB     7
F3C6  FE0A      0799          CP      10
F3C8  D8        0800          RET     C
F3C9  FE10      0801          CP      16
F3CB  3F        0802          CCF
F3CC  C9        0803          RET
                0804 ;
                0805 ;
                0806 ;
F3CD  7C        0807 PUT4HS: LD      A,H
F3CE  CDD8F3    0808          CALL    PUT2HX
F3D1  7D        0809          LD      A,L
F3D2  CDD8F3    0810 PUT2HS: CALL    PUT2HX
F3D5  C302F4    0811          JP      SPACE
                0812 ;
                0813 ;
F3D8  F5        0814 PUT2HX: PUSH    AF
F3D9  1F        0815          RRA
F3DA  1F        0816          RRA
F3DB  1F        0817          RRA
F3DC  1F        0818          RRA
F3DD  CDE1F3    0819          CALL    PUTNIB
F3E0  F1        0820          POP     AF
F3E1  E60F      0821 PUTNIB: AND     00001111B
```

```
F3F1  C9        0841          RET
                0842 ;
F3F2  7E        0843 PMSG:   LD      A,(HL)
F3F3  23        0844          INC     HL
F3F4  FE04      0845          CP      EOT
F3F6  C8        0846          RET     Z
F3F7  CD15F4    0847          CALL    OUTPUT
F3FA  18F6      0848          JR      PMSG-$
                0849 ;
                0850 ;
                0851 ;CRLFS OUTPUTS A RETURN-LINEFEED-SPACE
                0852 ;TO THE CONSOLE DEVICE
                0853 ;
F3FC  CDECF3    0854 CRLFS:  CALL    PNEXT
F3FF  0D0A04    0855          DEFB    CR,LF,EOT
F402  3E20      0856 SPACE:  LD      A,' '
F404  C315F4    0857          JP      OUTPUT
                0858 ;
                0859 ;
                0860 ;
                0861 ;ECHO INPUTS ONE CHARACTER FROM THE CONSOLE
                0862 ;DEVICE, PRINTS IT ON THE CONSOLE OUTPUT AND
                0863 ;THEN RETURNS IT IN REGISTER A WITH BIT 7 RESET
                0864 ;
                0865 ;OUTPUT PRINTS THE CHARACTER IN REGISTER A ON
                0866 ;THE CONSOLE OUTPUT DEVICE AND THEN DOES A CHECK
                0867 ;FOR CONSOLE INPUT TO FREEZE OR ABORT OUTPUT.
                0868 ;
                0869
F407  CD09F0    0870 ECHO:   CALL    CONIN       ;INPUT A CHARACTER AND ECHO IT
F40A  F5        0871          PUSH    AF
F40B  CD0CF0    0872          CALL    CONOUT
F40E  F1        0873          POP     AF
F40F  FE5B      0874          CP      'Z'+1
F411  D8        0875          RET     C
F412  D620      0876          SUB     32          ;CONVERT UPPER CASE TO LOWER
F414  C9        0877          RET
                0878 ;
                0879 ;
                0880 ;
F415  CD0CF0    0881 OUTPUT: CALL    CONOUT
F418  CD06F0    0882          CALL    CONST       ;SEE IF CONSOLE INPUT PENDING
F41B  280F      0883          JR      Z,OUTP2-$
F41D  CD09F0    0884          CALL    CONIN
F420  FE0D      0885          CP      CR          ;SEE IF <CR> WAS TYPED
F422  2805      0886          JR      Z,OUTP1-$
F424  CD09F0    0887          CALL    CONIN       ;WAIT FOR ANOTHER INPUT CHAR
F427  1803      0888          JR      OUTP2-$     ; THEN RET TO CALLING ROUTINE
                0889
F429  3284FF    0890 OUTP1:  LD      (ESCFLG),A  ;SET ESC FLAG TO NON-ZERO VALUE
F42C  3A84FF    0891 OUTP2:  LD      A,(ESCFLG)
F42F  B7        0892          OR      A           ;RETURN CURRENT STATUS OF ESC
F430  C9        0893          RET                 ; FLAG TO CALLING ROUTINE
                0894 ;
                0895 ;
                0896 ;
                0897          INCLUDE INTSRV.ASM
```

PFM Monitor Listing (continued)

```
                        0898 ;*************************************************************
                        0899 ;*                                                           *
                        0900 ;*       INTERRUPT SERVICE ROUTINES FOR KEYBOARD             *
                        0901 ;*       INPUT AND REAL-TIME CLOCK FUNCTIONS                 *
                        0902 ;*                                  3-Aug-80                 *
                        0903 ;*                                                           *
                        0904 ;*************************************************************
                        0905 ;
                        0906 ;
                        0907 ;
                        0908 ;
F431  3A30FF            0909 KBDST:  LD      A,(FIFCNT) ;GET INPUT FIFO BYTECOUNT
F434  B7                0910         OR      A          ;TEST IF EQUAL ZERO
F435  C8                0911         RET     Z          ;EXIT WITH A=0 IF QUEUE EMPTY
F436  3EFF              0912         LD      A,255
F438  C9                0913         RET                ;ELSE A=255 INDICATES DATA RDY
                        0914 ;
                        0915 ;
                        0916 ;
F439  CD31F4            0917 KBDIN:  CALL    KBDST
F43C  28FB              0918         JR      Z,KBDIN-$  ;LOOP UNTIL KEYBOARD INPUT RDY
F43E  E5                0919         PUSH    HL
F43F  CD6DF4            0920         CALL    REMOVE     ;GET CHARACTER FROM INPUT QUEUE
F442  E1                0921         POP     HL
F443  C9                0922         RET
                        0923 ;
                        0924 ;
                        0925 ;
                        0926 ;
                        0927 ;
F444  2133FF            0928 STASH:  LD      HL,LOCK    ;POINT TO SHIFT LOCK VARIABLES
F447  BE                0929         CP      (HL)       ;TEST IF A=SHIFT LOCK CHARACTER
F448  23                0930         INC     HL         ;THEN POINT TO LOCK FLAG
F449  2002              0931         JR      NZ,STASH2-$;JUMP IF NOT SHIFT CHARACTER
F44B  34                0932         INC     (HL)       ;ELSE COMPLIMENT THE SHIFT LOCK
F44C  C9                0933         RET                ;AND EXIT NOW
                        0934
F44D  CB46              0935 STASH2: BIT     0,(HL)     ;TEST THE SHIFT LOCK FLAG
F44F  280A              0936         JR      Z,STASH3-$ ;JUMP IF SHIFT LOCK NOT SET
F451  FE40              0937         CP      40H        ;ELSE CHECK FOR SHIFTABLE CHAR
F453  3806              0938         JR      C,STASH3-$ ;AND JUMP IF NOT = OR GREATER
F455  FE7F              0939         CP      7FH        ;THAN '@' AND LESS THAN RUBOUT
F457  3002              0940         JR      NC,STASH3-$
F459  EE20              0941         XOR     00100000B  ;ELSE TOGGLE BIT 5 OF THE CHAR
F45B  4F                0942 STASH3: LD      C,A
F45C  2130FF            0943         LD      HL,FIFCNT  ;BUMP INPUT FIFO CHAR COUNT
F45F  7E                0944         LD      A,(HL)
F460  3C                0945         INC     A
F461  FE10              0946         CP      16
F463  D0                0947         RET     NC         ;EXIT NOW IF FIFO IS FULL
F464  77                0948         LD      (HL),A     ;ELSE INCREMENT FIFO COUNT
F465  2131FF            0949         LD      HL,FIFIN   ;POINT HL TO FIFO INPUT OFFSET
F468  CD74F4            0950         CALL    INDEX
F46B  71                0951         LD      (HL),C     ;STORE CHARACTER IN FIFO @ HL
F46C  C9                0952         RET
                        0953 ;
                        0954 ;
                        0955 ;
                        0956 ;
```

```
F4C1  CDE7F4            1021 DSPTCH: CALL    CALLHL     ;CALL SUBROUTINE ADDRESSED BY H
F4C4  F1                1022         POP     AF
F4C5  C1                1023         POP     BC
F4C6  D1                1024         POP     DE
F4C7  E1                1025         POP     HL
F4C8  ED7B35FF          1026         LD      SP,(SPSAVE)
F4CC  FB                1027         EI                 ;RE-ENABLE INTERRUPTS & RETURN
F4CD  ED4D              1028         RETI
                        1029 ;
                        1030 ;
                        1031 ;-- RX ERROR INTERRUPT SERVICE ROUTINE FOR SIO --
                        1032 ;
                        1033 ;ARRIVE HERE IF RECEIVE INTERRUPT FROM FRAMING, OVERRUN
                        1034 ;AND PARITY ERRORS. (PARITY CAN BE DISABLED)
                        1035 ;
F4CF  ED7335FF          1036 SIOERR: LD      (SPSAVE),SP  ;SAVE USER STACK POINTER AND
F4D3  3157FF            1037         LD      SP,TMPSTK+32 ; SWITCH TO LOCAL STACK
F4D6  F5                1038         PUSH    AF
F4D7  CDF5F4            1039         CALL    SIOIN2     ;CLEAR BAD CHARACTER FROM SIO
F4DA  3E07              1040         LD      A,'G'-64
F4DC  CD15F5            1041         CALL    SIOXMT     ;OUTPUT A CTL-G AS A WARNING
F4DF  F1                1042         POP     AF
F4E0  ED7B35FF          1043         LD      SP,(SPSAVE)
F4E4  FB                1044         EI
F4E5  ED4D              1045         RETI
                        1046 ;
                        1047 ;
F4E7  E9                1048 CALLHL: JP      (HL)
                        1049 ;
                        1050 ;
                        1051 ;
                        1052 ;POLLED MODE I/O ROUTINES FOR SIO CHANEL B
                        1053 ;
F4E8  DB07              1054 SIOST:  IN      A,(SIOCPB) ;GET SIO STATUS REGISTER
F4EA  E601              1055         AND     00000001B
F4EC  C8                1056         RET     Z          ;ACC=0 IF NO DATA AVAILABLE
F4ED  3EFF              1057         LD      A,255
F4EF  C9                1058         RET
                        1059 ;
                        1060 ;
F4F0  CDE8F4            1061 SIOIN:  CALL    SIOST      ;TEST CONSOLE STATUS
F4F3  28FB              1062         JR      Z,SIOIN-$  ;LOOP UNTIL DATA IS RECEIVED
R4F5  3E30              1063 SIOIN2: LD      A,00110000B      ;RESET STATUS BITS IN SIO FO
F4F7  D307              1064         OUT     (SIOCPB),A ;PARITY/OVERRUN/FRAMING ERRORS,
F4F9  DB05              1065         IN      A,(SIODPB) ;THEN GET THE INPUT CHARACTER
F4FB  E67F              1066         AND     01111111B
F4FD  C9                1067         RET
                        1068 ;
                        1069 ;
F4FE  FE20              1070 SIOOUT: CP      ' '        ;TEST FOR CONTROL CHARACTERS
F500  3013              1071         JR      NC,SIOXMT-$;JUMP IF PRINTABLE CHARACTER
F502  CD15F5            1072         CALL    SIOXMT     ;ELSE SEND CONTROL CHARACTER
F505  3A79FF            1073         LD      A,(NULLS)  ;AND THEN SEND NULLS AS PADDING
F508  3C                1074         INC     A          ;GET NULL PAD COUNT AND FIX SO
F509  1806              1075         JR      PAD1-$     ;THAT COUNT=0 SENDS NO NULLS
                        1076
F50B  F5                1077 PAD:    PUSH    AF
F50C  AF                1078         XOR     A
F50D  CD15F5            1079         CALL    SIOXMT     ;OUTPUT A NULL TO THE SIO
F510  F1                1080         POP     AF
F511  3D                1081 PAD1:   DEC     A
F512  20F7              1082         JR      NZ,PAD-$   ;LOOP SENDING NULLS TO SIO
F514  C9                1083         RET
                        1084 ;
                        1085 ;
F515  F5                1086 SIOXMT: PUSH    AF
F516  DB07              1087 SIOX1:  IN      A,(SIOCPB)
```

```
F46D  2130FF    0957 REMOVE: LD      HL,FIFCNT
F470  35        0958         DEC     (HL)
F471  2132FF    0959         LD      HL,FIFOUT  ;POINT HL TO FIFO OUTPUT OFFSET
F474  7E        0960 INDEX:  LD      A,(HL)
F475  3C        0961         INC     A
F476  E60F      0962         AND     00001111B  ;INCREMENT FIFO POINTER
F478  77        0963         LD      (HL),A     ; MODULO 16 AND REPLACE
F479  2120FF    0964         LD      HL,FIFO
F47C  85        0965         ADD     A,L        ;INDEX INTO FIFO BY OFFSET IN A
F47D  6F        0966         LD      L,A
F47E  7E        0967         LD      A,(HL)
F47F  C9        0968         RET
                0969 ;
                0970 ;
                0971 ;SOFTWARE DISK MOTOR TURN-OFF TIMER ROUTINE
                0972 ;
F480  216CFF    0973 DSKTMR: LD      HL,MOTOR   ;DECREMENT DISK TURN-OFF TIMER
F483  35        0974         DEC     (HL)
F484  C0        0975         RET     NZ         ;EXIT IF NOT TIMED OUT YET
F485  DB1C      0976         IN      A,(BITDAT)
F487  F644      0977         OR      01000100B  ;DISABLE ALL DRIVE SELECTS AND
F489  D31C      0978         OUT     (BITDAT),A ; TURN OFF THE SPINDLE MOTORS
F48B  C9        0979         RET
                0980 ;
                0981 ;
                0982 ;-- INTERRUPT SERVICE ROUTINE FOR PARALLEL KEYBOARD --
                0983 ;
F48C  ED7335FF  0984 KEYSRV: LD      (SPSAVE),SP ;SAVE USR STACK POINT AND
F490  3157FF    0985         LD      SP,TMPSTK+32;SWITCH TO LOCAL STACK
F493  E5        0986         PUSH    HL
F494  D5        0987         PUSH    DE
F495  C5        0988         PUSH    BC
F496  F5        0989         PUSH    AF         ;SAVE MACHINE STATE
F497  DB1E      0990         IN      A,(KBDDAT) ;READ KEYBOARD INPUT PORT
F499  2F        0991         CPL
F49A  2A59FF    0992         LD      HL,(PINVEC);GET KBD INTERRUPT RTN VECTOR
F49D  1822      0993         JR      DSPTCH-$   ;AND JUMP TO DISPATCH POINT
                0994 ;
                0995 ;
                0996 ;
                0997 ;-- INTERRUPT SERVICE ROUTINE FOR ONE SECOND TIMER --
                0998 ;
F49F  ED7335FF  0999 TIMER:  LD      (SPSAVE),SP;SAVE USR STACK POINTER AND
F4A3  3157FF    1000         LD      SP,TMPSTK+32 ;SWITCH TO LOCAL STACK
F4A6  E5        1001         PUSH    HL
F4A7  D5        1002         PUSH    DE
F4A8  C5        1003         PUSH    BC
F4A9  F5        1004         PUSH    AF
F4AA  2A57FF    1005         LD      HL,(TIKVEC);GET CLOCK INTERRUPT RTN VECTOR
F4AD  1812      1006         JR      DSPTCH-$   ; AND JUMP TO DISPATCH POINT
                1007 ;
                1008 ;
                1009 ;
                1010 ;-- SERIAL INPUT INTERRUPT SERVICE ROUTINE FOR SIO --
                1011 ;
F4AF  ED7335FF  1012 SIOINT: LD      (SPSAVE),SP  ;SAVE USER STACK POINTER AND
F4B3  3157FF    1013         LD      SP,TMPSTK+32 ; SWITCH TO LOCAL STACK
F4B6  E5        1014         PUSH    HL
F4B7  D5        1015         PUSH    DE
F4B8  C5        1016         PUSH    BC
F4B9  F5        1017         PUSH    AF         ;SAVE MACHINE STATE
F4BA  DB05      1018         IN      A,(SIODPB) ;READ SIO DATA INPUT PORT
F4BC  E67F      1019         AND     01111111B
F4BE  2A5BFF    1020         LD      HL,(SINVEC);GET SERIAL INPUT RTN VECTOR
```


(continued on top of page 10)

```
F518  E604      1088         AND     00000100B  ;TEST TBE STATUS BIT
F51A  28FA      1089         JR      Z,SIOX1-$
F51C  F1        1090         POP     AF
F51D  D305      1091         OUT     (SIODPB),A ;OUTPUT DATA TO SIO
F51F  C9        1092         RET
                1093 ;
                1094 ;
                1095 ;
                1096 ;
                1097         INCLUDE CRTOUT.ASM
                1098 ;*******************************************************
                1099 ;*                                                     *
                1100 ;*       MEMORY-MAPPED CRT OUTPUT DRIVER               *
                1101 ;*                                                     *
                1102 ;*       Russell Smith     18-August-1980              *
                1103 ;*                                                     *
                1104 ;*******************************************************
                1105 ;
                1106 ;
>0030           1107 CRTBAS  EQU     CRTMEM.SHR.8 ;START PAGE# OF 3K CRT SPACE
>003C           1108 CRTTOP  EQU     CRTMEM+3072.SHR.8 ;END PAGE# OF CRT SPACE
                1109 ;
                1110 ;
F520  E5        1111 CRTOUT: PUSH    HL
F521  D5        1112         PUSH    DE
F522  C5        1113         PUSH    BC
F523  CBBF      1114         RES     7,A
F525  4F        1115         LD      C,A
F526  F3        1116         DI                 ;KEEP WOLVES AWAY FOR A WHILE
F527  ED7335FF  1117         LD      (SPSAVE),SP
F52B  3157FF    1118         LD      SP,TMPSTK+32 ;POINT SP TO TOP LOCAL STACK
F52E  DB1C      1119         IN      A,(BITDAT)
F530  CBFF      1120         SET     7,A        ;SELECT ROM/CRT MEMORY BANK
F532  D31C      1121         OUT     (BITDAT),A
                1122 ;
                1123 ;FIRST REMOVE THE OLD CURSOR CHARACTER FROM THE SCREEN
                1124 ;
F534  2175FF    1125         LD      HL,CHRSAV  ;GET CHAR OVERLAYED BY CURSOR
F537  46        1126         LD      B,(HL)
F538  2A73FF    1127         LD      HL,(CURSOR);LOAD HL WITH CURSOR POINTER
F53B  7C        1128         LD      A,H
F53C  E60F      1129         AND     00001111B  ;INSURANCE THAT HL CAN'T
F53E  F630      1130         OR      CRTBAS     ;EVER POINT OUTSIDE CRT MEMORY
F540  67        1131         LD      H,A
F541  70        1132         LD      (HL),B     ;RMV CURSOR BY RESTORING CHAR
                1133 ;
                1134 ;PROCESS CHARACTER PASSED IN C
                1135 ;
F542  CD65F5    1136         CALL    OUTCH
                1137 ;
                1138 ;NOW STORE A NEW CURSOR CHARACTER AT THE CURSOR LOCATION
                1139 ;
F545  7E        1140         LD      A,(HL)     ;GET CHAR AT NEW CURSOR LOCAT.
F546  3275FF    1141         LD      (CHRSAV),A ;SAVE FOR NEXT TIME 'CRTOUT' IS
;                                                CALLED
F549  FE20      1142         CP      ' '        ;TEST IF CHARACTER IS A SPACE
F54B  CBFF      1143         SET     7,A        ;THEN TURN ON BIT 7 TO ENABLE
;                                                BLINK
F54D  2003      1144         JR      NZ,CRT2-$  ;JUMP IF CHARACTER IS NON-BLANK
F54F  3A76FF    1145         LD      A,(CSRCHR) ;ELSE GET CHAR USED FOR CURSOR
F552  77        1146 CRT2:   LD      (HL),A     ;STORE CHAR IN A AS CURSOR MARK
F553  2273FF    1147         LD      (CURSOR),HL;SAVE HL AS CURSOR POINTER
                1148
F556  ED7B35FF  1149         LD      SP,(SPSAVE)
F55A  DB1C      1150         IN      A,(BITDAT)
```

(continued next page)

PFM Monitor Listing (continued)

```
F55C  CBBF      1151          RES     7,A         ;SWITCH BACK LOWER 16K OF RAM
F55E  D31C      1152          OUT     (BITDAT),A
F560  FB        1153          EI                  ;INTERRUPTS ARE SAFE AGAIN
F561  C1        1154          POP     BC
F562  D1        1155          POP     DE
F563  E1        1156          POP     HL
F564  C9        1157          RET
                1158 ;
                1159 ;
                1160 ;
F565  1178FF    1161 OUTCH:   LD      DE,LEADIN
F568  1A        1162          LD      A,(DE)      ;GET LEAD-IN SEQUENCE STATE
F569  B7        1163          OR      A
F56A  C270F6    1164          JP      NZ,MULTI    ;JUMP IF IN A LEAD-IN SEQUENCE
F56D  79        1165          LD      A,C         ; ELSE PROCESS CHARACTER IN C
F56E  FE20      1166          CP      ' '
F570  380F      1167          JR      C,CONTRL-$  ;JUMP IF A CONTROL CHARACTER
F572  71        1168 DISPLA:  LD      (HL),C      ;ELSE STORE DISPLAYABLE CHAR
F573  23        1169          INC     HL          ;AND ADV POINTER TO NEXT COLUMN
F574  7D        1170          LD      A,L
F575  E67F      1171          AND     01111111B   ;EXTRACT COLUMN# FROM HL
F577  FE50      1172          CP      80
F579  D8        1173          RET     C           ;EXIT IF NOT PAST COLUMN 79
F57A  CDE7F5    1174          CALL    RETURN      ;ELSE DO AUTOMATIC <CR>
F57D  CD42F6    1175          CALL    LFEED       ;AND LINEFEED
F580  C9        1176          RET
                1177 ;
                1178 ;
                1179 ;
F581  E5        1180 CONTRL:  PUSH    HL
F582  218FF5    1181          LD      HL,CTLTAB   ;SEARCH FOR CONTROL CHARACTER
F585  010D00    1182          LD      BC,CTLSIZ/3;HANDLING SUBROUTINE IN TABLE
F588  CD60F3    1183          CALL    SEARCH
F58B  E1        1184          POP     HL
F58C  C0        1185          RET     NZ          ;EXIT IF NOT IMPLEMENTED
F58D  C5        1186          PUSH    BC
F58E  C9        1187          RET                 ;DO SNEAKY JUMP TO PRESERVE
;                                                   REGISTERS
                1188
F58F  1F        1189 CTLTAB:  DEFB    '_'-64
F590  1E        1190          DEFB    '^'-64
F591  1B        1191          DEFB    '['-64
F592  1A        1192          DEFB    'Z'-64
F593  18        1193          DEFB    'X'-64
F594  11        1194          DEFB    'Q'-64
F595  0D        1195          DEFB    'M'-64
F596  0C        1196          DEFB    'L'-64
F597  0B        1197          DEFB    'K'-64
F598  0A        1198          DEFB    'J'-64
F599  09        1199          DEFB    'I'-64
F59A  08        1200          DEFB    'H'-64
F59B  07        1201          DEFB    'G'-64
                1202
F59C  DCF5      1203          DEFW    BELL        ;CTL-G IS THE BELL
F59E  BEF5      1204          DEFW    BAKSPC      ;CTL-H IS CURSOR LEFT
F5A0  CCF5      1205          DEFW    TAB         ;CTL-I IS TAB
F5A2  42F6      1206          DEFW    LFEED       ;CTL-J IS CURSOR DOWN
F5A4  2CF6      1207          DEFW    UPCSR       ;CTL-K IS CURSOR UP
F5A6  C4F5      1208          DEFW    FORSPC      ;CTL-L IS CURSOR RIGHT
```

```
F5EF  E5        1273          PUSH    HL
F5F0  110130    1274          LD      DE,CRTMEM+1
F5F3  01000C    1275          LD      BC,24*128
F5F6  3620      1276          LD      (HL),' '
F5F8  EDB0      1277          LDIR                ;FILL CRT MEMORY WITH SPACES
F5FA  E1        1278          POP     HL          ;POINT TO HOME CURSOR POSITION
F5FB  3E17      1279          LD      A,23
F5FD  3277FF    1280          LD      (BASE),A    ;MAKE BASE LINE# BE 23 AND
F600  D314      1281          OUT     (SCROLL),A  ; STORE IN SCROLL REGISTER
F602  C9        1282          RET
                1283 ;
                1284 ;
F603  E5        1285 CLREOL:  PUSH    HL          ;SAVE CURSOR POINTER
F604  7D        1286          LD      A,L
F605  E67F      1287          AND     01111111B   ;GET COLUMN# COMPONENT OF
F607  4F        1288          LD      C,A         ; CURSOR POINTER INTO C
F608  3E50      1289          LD      A,80        ;CALCULATE HOW MANY CHARS
F60A  91        1290          SUB     C           ; REMAIN ON CURRENT LINE
F60B  47        1291          LD      B,A
F60C  CD66F6    1292          CALL    CLR         ;CLEAR REST OF LINE @ HL
F60F  E1        1293          POP     HL
F610  C9        1294          RET
                1295 ;
                1296 ;
F611  CD03F6    1297 CLREOS:  CALL    CLREOL      ;CLEAR REMAINDER OF CURRENT ROW
F614  E5        1298          PUSH    HL
F615  3A77FF    1299          LD      A,(BASE)
F618  4F        1300          LD      C,A         ;COPY BASE SCREEN ROW# TO C
F619  7D        1301 CLRS1:   LD      A,L
F61A  17        1302          RLA
F61B  7C        1303          LD      A,H
F61C  17        1304          RLA                 ;ROW# COMPONENT OF HL INTO A
F61D  E61F      1305          AND     00011111B
F61F  B9        1306          CP      C           ;SEE IF HL IS AT BOTTOM ROW
;                                                   OF SCREEN
F620  2808      1307          JR      Z,CLRS2-$   ;AND LEAVE CLEAR LOOP IF SO
F622  CD37F6    1308          CALL    DNCSR       ;ELSE POINT HL TO NEXT ROW DOWN
F625  CD60F6    1309          CALL    CLRLIN      ;AND FILL THAT LINE WITH SPACES
F628  18EF      1310          JR      CLRS1-$
                1311
F62A  E1        1312 CLRS2:   POP     HL          ;RESTR ORIGINAL CURSOR POINTER
F62B  C9        1313          RET
                1314 ;
                1315 ;
F62C  1180FF    1316 UPCSR:   LD      DE,-128     ;SUBTRACT 1 FROM ROW# COMPONENT
F62F  19        1317          ADD     HL,DE       ; OF CURSOR POINTER IN HL
F630  7C        1318          LD      A,H
F631  FE30      1319          CP      CRTBAS      ;CHECK FOR UNDERFLOW OF POINTER
F633  D0        1320          RET     NC
F634  263B      1321          LD      H,CRTTOP-1  ;WRAP CURSOR AROUND MODULO 3K
F636  C9        1322          RET
                1323 ;
                1324 ;
F637  118000    1325 DNCSR:   LD      DE,128      ;ADD 1 TO ROW# COMPONENT
F63A  19        1326          ADD     HL,DE       ; OF CURSOR POINTER IN HL
F63B  7C        1327          LD      A,H
F63C  FE3C      1328          CP      CRTTOP      ;CHECK FOR OVERFLOW OF POINTER
F63E  D8        1329          RET     C
F63F  2630      1330          LD      H,CRTBAS    ;RESET POINTER MODULO 128*24
F641  C9        1331          RET
                1332 ;
                1333 ;
                1334 ;
F642  7D        1335 LFEED:   LD      A,L
F643  17        1336          RLA
F644  7C        1337          LD      A,H
```

```
F5A8  E7F5      1209            DEFW    RETURN      ;CTL-M IS <CR>
F5AA  11F6      1210            DEFW    CLREOS      ;CTL-Q CLEAR TO END-OF-SCREEN
F5AC  03F6      1211            DEFW    CLREOL      ;CTL-X IS CLEAR TO END-OF-LINE
F5AE  ECF5      1212            DEFW    CLRSCN      ;CTL-Z IS CLEAR SCREEN
F5B0  B6F5      1213            DEFW    ESCAPE      ;CTL-[ IS ESCAPE
F5B2  6CF6      1214            DEFW    HOMEUP      ;CTL-^ IS HOME UP
F5B4  BAF5      1215            DEFW    STUFF       ;CTL-_ IS DISPLAY CONTROL CHARS
                1216
>0027           1217 CTLSIZ     EQU     $-CTLTAB
                1218 ;
                1219 ;
F5B6  3E01      1220 ESCAPE:    LD      A,1
F5B8  12        1221            LD      (DE),A      ;SET LEAD-IN SEQUENCE STATE
F5B9  C9        1222            RET                 ;FOR XY CURSOR POSITIONING MODE
                1223 ;
                1224 ;
F5BA  3E04      1225 STUFF:     LD      A,4
F5BC  12        1226            LD      (DE),A      ;SET LEAD-IN SEQUENCE STATE
F5BD  C9        1227            RET                 ;FOR CONTROL CHAR OUTPUT MODE
                1228 ;
                1229 ;
F5BE  7D        1230 BAKSPC     LD      A,L         ;CHECK FOR LEFT MARGIN
F5BF  E67F      1231            AND     01111111B
F5C1  C8        1232            RET     Z           ;ABORT IF IN LEFTMOST COLUMN
F5C2  2B        1233            DEC     HL          ;BACK UP CURSOR POINTER
F5C3  C9        1234            RET
                1235 ;
                1236 ;
F5C4  7D        1237 FORSPC:    LD      A,L         ;CHECK FOR RIGHTMOST COLUNM
F5C5  E67F      1238            AND     01111111B
F5C7  FE4F      1239            CP      79
F5C9  D0        1240            RET     NC          ;DO NOTHING IF ALREADY THERE
F5CA  23        1241            INC     HL
F5CB  C9        1242            RET                 ;ELSE ADVANCE CURSOR POINTER
                1243 ;
                1244 ;
F5CC  110800    1245 TAB:       LD      DE,8        ;TABS ARE EVERY 8 COLUMNS
F5CF  7D        1246            LD      A,L         ;GET COLUMN COMPONENT OF
F5D0  E678      1247            AND     01111000B   ; PREVIOUS TAB POSITION
F5D2  83        1248            ADD     A,E
F5D3  FE50      1249            CP      80          ;EXIT IF NEXT TAB COLUMN WOULD
F5D5  D0        1250            RET     NC          ; BE PAST THE RIGHT MARGIN
F5D6  7D        1251            LD      A,L
F5D7  E6F8      1252            AND     11111000B   ;ELSE INCREMENT THE CURSOR
F5D9  6F        1253            LD      L,A         ; POINTER FOR REAL
F5DA  19        1254            ADD     HL,DE
F5DB  C9        1255            RET
                1256 ;
                1257 ;
F5DC  DB1C      1258 BELL:      IN      A,(BITDAT)
F5DE  CBEF      1259            SET     5,A         ;TOGGLE BIT 5 OF SYSTEM PIO TO
F5E0  D31C      1260            OUT     (BITDAT),A  ;TRIGGER BELL HARDWARE TO SOUND
F5E2  CBAF      1261            RES     5,A
F5E4  D31C      1262            OUT     (BITDAT),A
F5E6  C9        1263            RET
                1264 ;
                1265 ;
F5E7  7D        1266 RETURN:    LD      A,L
F5E8  E680      1267            AND     10000000B
F5EA  6F        1268            LD      L,A         ;MOVE CURSOR POINTER BACK
F5EB  C9        1269            RET                 ; TO START OF LINE
                1270 ;
                1271 ;
F5EC  210030    1272 CLRSCN:    LD      HL,CRTMEM
```

(continued on top of page 12)

```
F645  17        1338            RLA
F646  E61F      1339            AND     00011111B   ;EXTRACT ROW# COMPONENT OF HL
F648  4F        1340            LD      C,A         ;COPY ROW# TO C FOR SCROLL TEST
F649  CD37F6    1341            CALL    DNCSR       ;MOVE CURSOR TO NEXT ROW DOWN
F64C  3A77FF    1342            LD      A,(BASE)    ;TEST IF CURSOR ON BOTTOM ROW
F64F  B9        1343            CP      C           ;OF SCREEN BEFORE MOVING DOWN
F650  C0        1344            RET     NZ          ;EXIT IF NOT AT BOTTOM
                1345
F651  E5        1346            PUSH    HL          ;ELSE PREP TO SCROLL SCREEN UP
F652  CD60F6    1347            CALL    CLRLIN      ;FILL NEW BOTTOM LINE WTH SPACES
F655  29        1348            ADD     HL,HL
F656  7C        1349            LD      A,H         ;GET ROW# PART OF HL INTO A
F657  E61F      1350            AND     00011111B
F659  3277FF    1351            LD      (BASE),A    ;STORE NEW BASE LINE#
F65C  D314      1352            OUT     (SCROLL),A  ;SCROLL UP NEW BLANK BOTTM LINE
F65E  E1        1353            POP     HL
F65F  C9        1354            RET
                1355 ;
                1356 ;
F660  7D        1357 CLRLIN:    LD      A,L
F661  E680      1358            AND     10000000B   ;POINT HL TO 1ST COLUMN OF ROW
F663  6F        1359            LD      L,A
F664  0650      1360            LD      B,80
F666  3620      1361 CLR:       LD      (HL),' '    ;STORE ASCII SPACES AT ADDR
                                                     IN HL
F668  23        1362            INC     HL          ;AND INCREMENT HL
F669  10FB      1363            DJNZ    CLR-$       ;REPEAT NUMBER OF TIMES IN B
F66B  C9        1364            RET
                1365 ;
                1366 ;
F66C  0E20      1367 HOMEUP:    LD      C,' '       ;FAKE-OUT CURSOR ADDR ROUTINE
F66E  1817      1368            JR      SETROW-$    ;TO DO HOMEUP ALMOST FOR FREE
                1369 ;
                1370 ;
F670  EB        1371 MULTI:     EX      DE,HL       ;UNCONDITIONALLY RESET LEAD-IN
F671  3600      1372            LD      (HL),0      ;STATE TO ZERO BEFORE GOING ON
F673  EB        1373            EX      DE,HL
F674  FE01      1374            CP      1
F676  2008      1375            JR      NZ,M2TST-$
F678  79        1376 SETXY:     LD      A,C         ;GET SECOND CHAR OF SEQUENCE
F679  FE3D      1377            CP      '='
F67B  C0        1378            RET     NZ          ;ABORT SEQUENCE IF NOT '='
F67C  3E02      1379            LD      A,2
F67E  12        1380            LD      (DE),A      ;MAKE LEADIN=2 NEXT TIME
F67F  C9        1381            RET
                1382
F680  FE02      1383 M2TST:     CP      2
F682  2019      1384            JR      NZ,M3TST-$
F684  3E03      1385            LD      A,3
F686  12        1386            LD      (DE),A      ;MAKE LEADIN=3 NEXT TIME
F687  3A77FF    1387 SETROW:    LD      A,(BASE)    ;ARRIVE HERE ON THIRD CHAR
F68A  81        1388            ADD     A,C         ; OF ESC,'=',ROW,COL SEQUENCE
F68B  D61F      1389            SUB     ' '-1
F68D  D618      1390 SETR2:     SUB     24
F68F  30FC      1391            JR      NC,SETR2-$  ;VERIFY ROW# BETWEEN 0 AND 23
F691  C618      1392            ADD     A,24
F693  F660      1393            OR      CRTMEM.SHR.7 ;MERGE IN MSB'S OF CRT MEMORY
F695  67        1394            LD      H,A
F696  2E00      1395            LD      L,0
F698  CB3C      1396            SRL     H
F69A  CB1D      1397            RR      L
F69C  C9        1398            RET
                1399
F69D  FE03      1400 M3TST:     CP      3
F69F  200C      1401            JR      NZ,M4TST-$
```

(continued next page)

PFM Monitor Listing (continued)

```
F6A1  79        1402 SETCOL: LD      A,C          ;ARRIVE HERE ON FOURTH CHAR
F6A2  D620      1403         SUB     ' '          ;OF ESC,'=',ROW,COL SEQUENCE
F6A4  D650      1404 SETC2:  SUB     80
F6A6  30FC      1405         JR      NC,SETC2-$ ;MAKE SURE COL# BETWEEN 0 & 79
F6A8  C650      1406         ADD     A,80
F6AA  B5        1407         OR      L            ;MERGE IN COL# WITH L
F6AB  6F        1408         LD      L,A
F6AC  C9        1409         RET
                1410
F6AD  CD72F5    1411 M4TST:  CALL    DISPLA       ;DISPLAY THE CONTROL CHAR
F6B0  C9        1412         RET                  ;PASSED IN C
                1413 ;
                1414 ;
                1415 ;
                1416 ;
                1417         INCLUDE DISKIO.ASM
                1418 ;*******************************************************
                1419 ;*                                                     *
                1420 ;*      DISK INPUT/OUTPUT DRIVER SUBROUTINE PACKAGE    *
                1421 ;*      FOR WESTERN DIGITAL 1771 DISK CONTROLLER       *
                1422 ;*                                                     *
                1423 ;*      bullet-proof error recovery added 12-APR-80    *
                1424 ;*                                                     *
                1425 ;*******************************************************
                1426 ;
                1427 ;
                1428 ;EQUATES FOR DISK CONTROLLER PORTS AND COMMAND CODES
                1429 ;
>0010           1430 STSREG  EQU     WD1771+0     ;STATUS REGISTER
>0010           1431 CMDREG  EQU     WD1771+0     ;COMMAND REGISTER
>0011           1432 TRKREG  EQU     WD1771+1     ;TRACK REGISTER
>0012           1433 SECREG  EQU     WD1771+2     ;SECTOR REGISTER
>0013           1434 DATREG  EQU     WD1771+3     ;DATA REGISTER
                1435 ;
>0088           1436 RDCMD   EQU     10001000B    ;READ COMMAND
>00A8           1437 WRTCMD  EQU     10101000B    ;WRITE COMMAND
>001C           1438 SKCMD   EQU     00011100B    ;SEEK COMMAND
>00D0           1439 FINCMD  EQU     11010000B    ;FORCE INTR COMMAND
>000C           1440 RSTCMD  EQU     00001100B    ;RESTORE COMMAND
>0004           1441 HLOAD   EQU     00000100B    ;RD/WRT HEAD LOAD ENABLE
                1442 ;
>00C9           1443 RET     EQU     0C9H         ;SUBROUTINE RETURN INSTR OPCODE
>0066           1444 NMIVEC  EQU     0066H        ;THE NON-MASKABLE INTERRUPT IS
                1445                              ;USED FOR DATA SYNC BETWEEN
                1446                              ;THE Z-80 AND 1771
                1447 ;
                1448 ;
                1449 ;
F6B1  79        1450 SELECT: LD      A,C          ;GET UNIT# PASSED IN C AND
F6B2  FE04      1451         CP      4            ; CHECK FOR MAXIMUM VALID#
F6B4  D0        1452         RET     NC           ;ERROR IF NUMBER > 3
F6B5  CDB8F7    1453         CALL    TURNON       ;MAKE SURE DISKS ARE TURNED ON
F6B8  DB1C      1454         IN      A,(BITDAT)
F6BA  47        1455         LD      B,A          ;SAVE CURRENT DRIVE SELECT DATA
F6BB  E6F8      1456         AND     11111000B    ;MERGE IN NEW DRIVE UNIT# IN C
F6BD  B1        1457         OR      C            ;IN PLACE OF THE CURRENT ONE
F6BE  D31C      1458         OUT     (BITDAT),A ;TO SELECT THE NEW DISK DRIVE
F6C0  CDAEF7    1459         CALL    FORCE        ;TEST NEW DRIVE'S READY STATUS
F71F  CDABF7    1524 WRITE:  CALL    READY        ;CLEAR THE DISK CONTROLLER
F722  C0        1525         RET     NZ           ;EXIT IF DRIVE NOT READY
F723  CB77      1526         BIT     6,A
F725  C0        1527         RET     NZ           ;EXIT IF DISK WRITE-PROTECTED
F726  06A8      1528         LD      B,WRTCMD
F728  1806      1529         JR      RDWRT-$
                1530
F72A  CDABF7    1531 READ:   CALL    READY        ;CLEAR DISK CONTROLLER
F72D  C0        1532         RET     NZ           ;EXIT IF DRIVE NOT READY
F72E  0688      1533         LD      B,RDCMD
F730  2271FF    1534 RDWRT:  LD      (IOPTR),HL ;STORE DISK I/O DATA POINTER
F733  216EFF    1535         LD      HL,SECTOR
F736  71        1536         LD      (HL),C       ;STORE SECTOR# FOR READ/WRITE
F737  23        1537         INC     HL
F738  70        1538         LD      (HL),B       ;SAVE READ/WRITE COMMAND BYTE
F739  23        1539         INC     HL
F73A  3602      1540         LD      (HL),2       ;SET DISK RE-TRY COUNT
F73C  F3        1541 RW1:    DI                   ;NO INTERRUPTS DURING DISK I/O
F73D  216600    1542         LD      HL,NMIVEC    ;SAVE BYTE AT NMI VECTOR LOCAT
F740  56        1543         LD      D,(HL)       ;IN D FOR DURATION OF READ/WRIT
F741  36C9      1544         LD      (HL),RET     ;LOOP AND REPLACE IT WITH A RET
F743  216BFF    1545         LD      HL,RECLEN
F746  46        1546         LD      B,(HL)       ;B=NUMBER OF BYTES/SECTOR
F747  0E13      1547         LD      C,DATREG     ;C=1771 DATA REGISTER PORT#
F749  2A71FF    1548         LD      HL,(IOPTR)   ;HL=DISK R/W DATA POINTER
F74C  3A6EFF    1549         LD      A,(SECTOR)   ;GET SECTOR NUMBER
F74F  D312      1550         OUT     (SECREG),A ;OUTPUT SECTOR# TO 1771
F751  CDAEF7    1551         CALL    FORCE        ;ISSUE FORCE INTERRUPT COMMAND
F754  CB6F      1552         BIT     5,A          ;TO TEST HEAD LOAD STATUS
F756  3A6FFF    1553         LD      A,(CMDTYP)   ;GET READ OR WRITE COMMAND BYTE
F759  2002      1554         JR      NZ,RW2-$     ;JUMP IF HEAD IS ALREADY LOADED
F75B  F604      1555         OR      HLOAD        ;ELSE MERGE IN HLD BIT
F75D  CDA3F7    1556 RW2:    CALL    CMDOUT       ;START 1771 DOING IT'S THING
F760  CB6F      1557         BIT     5,A          ;TEST IF COMMAND IS A R OR W
F762  200D      1558         JR      NZ,WLOOP-$ ;AND JUMP TO THE CORRECT LOOP
F764  76        1559 RLOOP:  HALT
F765  EDA2      1560         INI
F767  C264F7    1561         JP      NZ,RLOOP
F76A  CD9CF7    1562         CALL    BUSY         ;LOOP UNTIL 1771 COMES UN-BUSY
F76D  E69C      1563         AND     10011100B  ;MASK OFF TO READY,NOT FOUND,CRC
F76F  180B      1564         JR      RW3-$        ;AND LOST DATA STATUS BITS
                1565
F771  76        1566 WLOOP:  HALT
F772  EDA3      1567         OUTI
F774  C271F7    1568         JP      NZ,WLOOP
F777  CD9CF7    1569         CALL    BUSY
F77A  E6BC      1570         AND     10111100B  ;MASK OFF AS ABOVE + WRT FAULT
F77C  216600    1571 RW3:    LD      HL,NMIVEC
F77F  72        1572         LD      (HL),D       ;RESTORE BYTE @ NMI VECTOR
F780  FB        1573         EI
F781  C8        1574         RET     Z            ;RETURN IF NO DISK I/O ERRORS
F782  2170FF    1575         LD      HL,RETRY
F785  35        1576         DEC     (HL)         ;DECREMENT RE-TRY COUNT AND
F786  2002      1577         JR      NZ,RW4-$     ; EXECUTE COMAND AGAIN IF NOT=0
F788  B7        1578         OR      A
F789  C9        1579         RET                  ;ELSE RETURN 1771 ERROR STATUS
                1580
F78A  216DFF    1581 RW4:    LD      HL,TRACK
F78D  4E        1582         LD      C,(HL)       ;GET TRACK# FOR THIS OPERATION
F78E  CDFBF6    1583         CALL    SEEK         ;TRY TO RE-CAILBRATE THE HEAD
F791  18A9      1584         JR      RW1-$        ;BEFORE READ OR WRITE AGAIN
                1585 ;
                1586 ;
                1587 ;
```

```
F6C3  2806      1460          JR      Z,SEL2-$    ;AND CONTINUE IF ITS READY
F6C5  78        1461          LD      A,B
F6C6  D31C      1462          OUT     (BITDAT),A  ;ELSE PUT BACK OLD DRIVE SELECT
F6C8  3E80      1463          LD      A,10000000B;AND RETURN DRIVE-NOT-READY
F6CA  C9        1464          RET
                1465
F6CB  2165FF    1466 SEL2:    LD      HL,UNIT     ;POINT HL TO DRIVE SELECT DATA
F6CE  7E        1467          LD      A,(HL)      ;LOAD A WITH CURRENT UNIT#
F6CF  71        1468          LD      (HL),C      ;AND STORE NEW UNIT# FROM C
F6D0  FEFF      1469          CP      255         ;TEST IF NO DRIVE SELECTED
F6D2  2806      1470          JR      Z,SEL3-$    ;YET & SKIP NEXT SEGMENT IF SO
F6D4  23        1471          INC     HL          ;POINT TO HEAD POSITION TABLE
F6D5  85        1472          ADD     A,L         ;AND ADD IN NEW UNIT# AS INDEX
F6D6  6F        1473          LD      L,A
F6D7  DB11      1474          IN      A,(TRKREG)  ;GET CURRENT HEAD POSITION
F6D9  77        1475          LD      (HL),A      ;AND STORE IN TABLE @ HL
F6DA  2166FF    1476 SEL3:    LD      HL,TRKTAB
F6DD  7D        1477          LD      A,L
F6DE  81        1478          ADD     A,C         ;INDEX INTO TABLE TO GET
F6DF  6F        1479          LD      L,A         ;HEAD POSITION OF NEW DRIVE
F6E0  7E        1480          LD      A,(HL)
F6E1  FEFF      1481          CP      255         ;TEST IF NEW DRIVE WAS EVER
F6E3  2804      1482          JR      Z,HOME-$    ;SELECTED AND DO A HOME IF NOT
F6E5  D311      1483          OUT     (TRKREG),A  ;OUTPUT DRIVE'S CURRENT HEAD
F6E7  AF        1484          XOR     A           ;POSITION TO THE TRACK REGISTER
F6E8  C9        1485          RET
                1486 ;
                1487 ;
                1488 ;
F6E9  CDABF7    1489 HOME:    CALL    READY       ;CLEAR DISK CONTROLLER
F6EC  C0        1490          RET     NZ          ;EXIT IF DRIVE NOT READY
F6ED  AF        1491          XOR     A
F6EE  326DFF    1492          LD      (TRACK),A   ;SET TRACK# IN MEM TO ZERO
F6F1  060C      1493 RESTOR:  LD      B,RSTCMD    ;LOAD B WITH A RESTORE COMMAND
F6F3  CD93F7    1494          CALL    STEP        ;EXECUTE HEAD MOVING OPERATION
F6F6  EE04      1495          XOR     00000100B   ;GET TRUE TRACK 0 STATUS
F6F8  E69C      1496          AND     10011100B   ;MASK TO ERROR BITS
F6FA  C9        1497          RET                 ;RETURN 1771 STATUS IN A
                1498 ;
                1499 ;
                1500 ;
F6FB  CDABF7    1501 SEEK:    CALL    READY       ;CLEAR DISK CONTROLLER
F6FE  C0        1502          RET     NZ          ;EXIT IF DRIVE NOT READY
F6FF  79        1503          LD      A,C         ;GET TRACK# DATA FROM C AND
F700  FE4D      1504          CP      77          ;CHECK FOR MAXIMUM VALID#
F702  D0        1505          RET     NC          ;FORGET IT IF TRACK# > 76
F703  326DFF    1506          LD      (TRACK),A   ;ELSE STORE TRACK# FOR SEEK
F706  D313      1507          OUT     (DATREG),A  ;OUTPUT TRACK # TO 1771
F708  061C      1508          LD      B,SKCMD     ;LOAD B WITH A SEEK COMMAND AND
F70A  CD93F7    1509          CALL    STEP        ;GO SEEK WITH PROPER STEP RATE
F70D  E698      1510          AND     10011000B   ;MASK TO READY,SEEK & CRC ERROR
F70F  C8        1511          RET     Z           ;BITS AND RETURN IF ALL GOOD
                1512
F710  CDF1F6    1513          CALL    RESTOR      ;ELSE TRY TO RE-CAILBRATE HEAD
F713  C0        1514          RET     NZ          ;ERROR IF WE CAN'T FIND TRACK 0
F714  79        1515          LD      A,C
F715  D313      1516          OUT     (DATREG),A  ;OUTPUT TRACK# TO 1771
F717  061C      1517          LD      B,SKCMD
F719  CD93F7    1518          CALL    STEP        ;TRY TO SEEK THE TRACK AGAIN
F71C  E698      1519          AND     10011000B
F71E  C9        1520          RET                 ;RETURN FINAL SEEK STATUS IN A
                1521 ;
                1522 ;
                1523 ;
```


(continued on top of page 14)

```
F793  3A6AFF    1588 STEP:    LD      A,(SPEED)   ;GET STEP SPEED VARIABLE
F796  E603      1589          AND     00000011B
F798  B0        1590          OR      B           ;MERGE WTH SEEK/HOME COMND IN B
F799  CDA3F7    1591          CALL    CMDOUT      ;OUTPUT COMMAND AND DELAY
F79C  DB10      1592 BUSY:    IN      A,(STSREG)
F79E  CB47      1593          BIT     0,A         ;TEST BUSY BIT FROM
F7A0  20FA      1594          JR      NZ,BUSY-$   ; 1771 AND LOOP TILL=0
F7A2  C9        1595          RET
                1596 ;
                1597 ;
                1598 ;
F7A3  D310      1599 CMDOUT:  OUT     (CMDREG),A  ;OUTPUT A COMMAND TO THE 1771
F7A5  CDA8F7    1600          CALL    PAUSE       ;WASTE 44 MICROSECONDS
F7A8  E3        1601 PAUSE:   EX      (SP),HL
F7A9  E3        1602          EX      (SP),HL
F7AA  C9        1603          RET
                1604 ;
                1605 ;
                1606 ;
F7AB  CDB8F7    1607 READY:   CALL    TURNON      ;KEEP THOSE DISKS SPINING FOLKS
F7AE  3ED0      1608 FORCE:   LD      A,FINCMD    ;ISSUE FORCE INTERRUPT COMMAND
F7B0  CDA3F7    1609          CALL    CMDOUT
F7B3  DB10      1610          IN      A,(STSREG)  ;READ STATUS REGISTER CONTENTS
F7B5  CB7F      1611          BIT     7,A         ;TEST DRIVE NOT READY BIT
F7B7  C9        1612          RET
                1613 ;
                1614 ;
                1615 ;
F7B8  3E1E      1616 TURNON:  LD      A,30
F7BA  326CFF    1617          LD      (MOTOR),A   ;RE-LOAD MOTOR TURN-OFF TIMER
F7BD  CDA8F7    1618          CALL    PAUSE
F7C0  DB1C      1619          IN      A,(BITDAT)
F7C2  CB57      1620          BIT     2,A         ;TEST IF MOTORS HAVE STOPPED
F7C4  C8        1621          RET     Z           ;AND EXIT IF STILL TURNED ON
F7C5  E6BB      1622          AND     10111011B   ;ELSE RE-ENABLE DRIVE SELECTS
F7C7  D31C      1623          OUT     (BITDAT),A  ;AND ACTIVATE THE MOTOR RELAY
F7C9  C5        1624          PUSH    BC
F7CA  0600      1625          LD      B,0         ;SET READY LOOP MAX TIMEOUT
F7CC  CDDCF7    1626 TURN2:   CALL    WAIT        ;WAIT 1/93 SECOND & TEST READY
F7CF  2802      1627          JR      Z,TURN3-$   ;EXIT LOOP IF DRIVE READY
F7D1  10F9      1628          DJNZ    TURN2-$     ;ELSE TRY AGAIN UP TO 256 TIMES
F7D3  0609      1629 TURN3:   LD      B,9
F7D5  CDDCF7    1630 TURN4:   CALL    WAIT        ;GIVE ABT 1/10 SEC MORE DELAY
F7D8  10FB      1631          DJNZ    TURN4-$
F7DA  C1        1632          POP     BC
F7DB  C9        1633          RET
                1634 ;
                1635 ;
F7DC  DB1B      1636 WAIT:    IN      A,(CTC3)    ;GET CURRENT CTC3 COUNT VALUE
F7DE  4F        1637          LD      C,A
F7DF  DB1B      1638 WAIT2:   IN      A,(CTC3)
F7E1  B9        1639          CP      C           ;SEE IF CTC3 CHANGED BY 1 COUNT
F7E2  28FB      1640          JR      Z,WAIT2-$   ;AND LOOP UNTIL IT CHANGES
F7E4  18C8      1641          JR      FORCE-$     ;THEN TEST DRIVE READY STATUS
                1642 ;
                1643 ;
                1644 ;
                1645 ;
                1646 ;
F7E6  0000      1647 ROMEND:  DEFW    0           ;TAIL OF FREE MEM LINKED LIST
                1648 ;
>FF00           1649          ORG     RAM
                1650          INCLUDE MEMORY.ASM
```

(continued next page)

PFM Monitor Listing (continued)

```
                1651 ;*************************************************************
                1652 ;*                                                           *
                1653 ;*       STORAGE ALLOCATION FOR 256 BYTE SCRATCH RAM          *
                1654 ;*                                                           *
                1655 ;*************************************************************
                1656 ;
                1657 ;
                1658
>FF00           1659 VECTAB  EQU     $           ;INTERRUPT VECTOR TABLE STARTS
>FF00           1660 SIOVEC: DEFS    16          ;SPACE FOR 8 VECTORS FOR SIO
>FF10           1661 CTCVEC: DEFS    8           ;SPACE FOR 4 VECTORS FOR CTC
>FF18           1662 SYSVEC: DEFS    4           ;SPACE FOR 2 VECTORS FOR SYSTEM
;                                                  PIO
>FF1C           1663 GENVEC: DEFS    4           ;SPACE FOR 2 VECTORS FOR
;                                                  GENERAL PIO
                1664 ;
                1665 ;
                1666 ;KEYBOARD DATA INPUT FIFO VARIABLES
                1667
>FF20           1668 FIFO:   DEFS    16          ;CONSOLE INPUT FIFO
>FF30           1669 FIFCNT: DEFS    1           ;FIFO DATA COUNTER
>FF31           1670 FIFIN:  DEFS    1           ;FIFI INPUT POINTER
>FF32           1671 FIFOUT: DEFS    1           ;FIFO OUTPUT POINTER
>FF33           1672 LOCK:   DEFS    2           ;SHIFT LOCK CHAR+FLAG BYTE
                1673 ;
                1674 ;
                1675 ;STACK POINTER SAVE AND LOCAL STACK FOR INTERRUPT ROUTINES
                1676
>FF35           1677 SPSAVE: DEFS    2           ;USER STACK POINTER SAVE AREA
>FF37           1678 TMPSTK: DEFS    32          ;LOCAL STACK FOR INTERRUPTS
                1679 ;
                1680 ;
                1681 ;'SOFTWARE' VECTORS FOR INTERRUPT SERVICE ROUTINES
                1682
>FF57           1683 TIKVEC: DEFS    2           ;1 SEC INTERRUPT ROUTINE VECTOR
>FF59           1684 PINVEC: DEFS    2           ;PARALLEL CONSOLE INPUT VECTOR
>FF5B           1685 SINVEC: DEFS    2           ;SERIAL CONSOLE INPUT VECTOR
                1686 ;
                1687 ;
                1688 ;CLOCK-TIMER INTERRUPT VARIABLES
                1689
>FF5D           1690 TIKCNT: DEFS    2           ;BINARY CLOCK TICK COUNTER
>FF5F           1691 DAY:    DEFS    1           ;CALENDAR DAY
>FF60           1692 MONTH:  DEFS    1           ;  MONTH
>FF61           1693 YEAR:   DEFS    1           ;  YEAR
>FF62           1694 HRS:    DEFS    1           ;CLOCK HOURS REGISTER
>FF63           1695 MINS:   DEFS    1           ;      MINUTES RETISTER
>FF64           1696 SECS:   DEFS    1           ;      SECONDS REGISTER
                1697 ;
                1698 ;
                1699 ;DISK I/O DRIVER VARIABLES
                1700
>FF65           1701 UNIT:   DEFS    1           ;CURRENTLY SELECTED DISK#
>FF66           1702 TRKTAB: DEFS    4           ;4 DRIVE HEAD POSITION TABLE
>FF6A           1703 SPEED:  DEFS    1           ;SEEK SPEED FOR 1771 COMMANDS
>FF6B           1704 RECLEN: DEFS    1           ;SECTOR RECORD LENGTH VARIABLE
>FF6C           1705 MOTOR:  DEFS    1           ;DRIVE MOTOR TURN-OFF TIMER
>FF6D           1706 TRACK:  DEFS    1
>FF6E           1707 SECTOR: DEFS    1
>FF6F           1708 CMDTYP: DEFS    1           ;COMMAND BYTE FOR READS/WRITES
>FF70           1709 RETRY:  DEFS    1           ;DISK OPERATION RE-TRY COUNT
>FF71           1710 IOPTR:  DEFS    2           ;DISK I/O BUFFER POINTER
                1711 ;
                1712 ;
                1713 ;
                1714 ;CRT OUTPUT DRIVER VARIABLES
                1715
>FF73           1716 CURSOR: DEFS    2           ;CURSOR POINTER
>FF75           1717 CHRSAV: DEFS    1           ;CHAR OVERLAYED BY CURSOR
>FF76           1718 CSRCHR: DEFS    1           ;CHAR USED FOR A CURSOR
>FF77           1719 BASE:   DEFS    1           ;CURRENT CONTENTS OF SCROLL
;                                                  REGISTER
>FF78           1720 LEADIN: DEFS    1           ;STATE OF LEAD-IN SEQUENCE
;                                                  HANDLER
                1721 ;
                1722 ;
                1723 ;NULL PAD COUNT FOR SERIAL OUTPUT DELAY
                1724
>FF79           1725 NULLS:  DEFS    1           ;# OF NULLS SENT AFTER CONTROL
;                                                  CHARS.
                1726 ;
                1727 ;
                1728 ;LISTHEAD POINTER FOR DYNAMIC MEMORY ALLOCATION SCHEME
                1729
>FF7A           1730 FREPTR: DEFS    2
                1731 ;
                1732 ;
                1733 ;CONSOLE MONITOR PROGRAM VARIABLES
                1734
>FF7C           1735 PARAM1: DEFS    2           ;STORAGE FOR NUMBERS READ
>FF7E           1736 PARAM2: DEFS    2           ;FROM LINE INPUT BUFFER
>FF80           1737 PARAM3: DEFS    2           ;BY 'PARAMS' SUBROUTINE
>FF82           1738 PARAM4: DEFS    2
>FF84           1739 ESCFLG: DEFS    1           ;CONSOLE ESCAPE FLAG
>FF85           1740 COFLAG: DEFS    1           ;CONSOLE OUTPUT TOGGLE
>FF86           1741 LAST:   DEFS    2           ;LAST ADDRESS USED BY 'MEMDMP'
>FF88           1742 LINBUF: DEFS    64          ;CONSOLE LINE INPUT BUFFER
                1743 ;
                1744 ;
                1745
                1746 ;
                1747         END

ERRORS=0000
```

end

MICRO CORNUCOPIA

Journal of the Big Board Users Group

P.O. BOX 223
BEND, OREGON 97709